A RAND NOTE

An Intelligent Information Dictionary for Semantic Manipulation of Relational Databases

Stephanie J. Cammarata

March 1988

40 Years
1948-1988
RAND

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under RAND's National Defense Research Institute, a Federally Funded Research and Development Center supported by the Office of the Secretary of Defense, Contract No. MDA903-85-C-0030.

This Note contains an offprint of RAND research originally published in a journal or book. The text is reproduced here, with permission of the original publisher.

Published by The RAND Corporation
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

A RAND NOTE

N-2860-DARPA

An Intelligent Information Dictionary for Semantic Manipulation of Relational Databases

Stephanie J. Cammarata

March 1988

Prepared for
The Defense Advanced Research Projects Agency

40 Years 1948-1988

RAND

An Intelligent Information Dictionary for Semantic Manipulation of Relational Databases[1]

Stephanie J. Cammarata
The RAND Corporation
1700 Main Street
Santa Monica, CA
90406-2138

ABSTRACT

This paper describes an intelligent information dictionary (IID) which serves as a knowledge-based interface between a database user and the query language of a relational database management system. IID extends the traditional roles of a data dictionary by enabling a user to view, manipulate, and verify semantic aspects of relational data. Our use of IID focuses on the interactive creation of simulation-specific databases from large "public" databases in the domain of military simulation and modeling. We have identified classes of database-related activities performed by a simulation developer when preparing databases as input to simulation models. Three categories of IID capabilities supporting these activities are: *explanation and browsing*, *customized data manipulation*, and *interactive consistency checking*. In this paper we detail specific features of these categories and present examples of their use.

1. Introduction

An intelligent information dictionary extends the traditional roles of a data dictionary by enabling the user to view, manipulate, and verify semantic aspects of data not expressed in a relational database. In the past, data dictionary systems have served as an interface between the database management system (DBMS) and the application programs that access the data. This close coupling of data dictionary, DBMS, and application programs excludes facilities for interactive access by a casual user. With the advent of workstation environments, interactive software, and public domain databases, the use of DBMS is no longer limited to database administrators (DBAs), operations managers, and application programs. Researchers and practitioners in many disciplines are experimenting with DBMS for organizing, maintaining, and sharing databases [McCa82]. Unfortunately, the DBMS tools and languages currently in

[1] This research was sponsored by the Defense Advanced Research Projects Agency under the auspices of RAND's National Defense Research Institute, a Federally Funded Research and Development Center sponsored by the Office of the Secretary of Defense.

Reprinted from *Advances in Database Technology—EDBT '88* International Conference on Extending Database Technology, Venice, Italy, March 1988, pp. 214–230, © Springer-Verlag. Reprinted by permission.

existence do not have facilities to aid these users in understanding and accessing the information they need [Curt81].

In this paper we discuss an intelligent information dictionary (IID) system which we developed as a knowledge-based interface between an interactive user and the query language (QL) of a relational DBMS. IID aids a user in understanding the organization of a relational database by providing application-specific explanations of relations, domains, attributes, and constraints. This facility combines knowledge of the domain with knowledge of relational database concepts to produce interactive tools for browsing, customized data manipulation, and interactive consistency checking. IID encourages users to interact with a relational database by manipulating semantic entities and relationships which are implicit in the relational representation.

In the next section we present a scenario that motivates this research within the domain of military simulation and modeling. We describe three categories of database related activities performed by a simulation developer when preparing databases as input to simulation models. Section 3 discusses IID capabilities supporting these interactive database "preparation" activities and presents examples of their use. The architecture of IID is outlined in Section 4, and Section 5 discusses related research. In Section 6 we conclude with some directions for future work.

2. Motivation and rationale

National security policy research and analysis depends on the heavy use of military modeling and simulation, such as battle management, and command and control studies. It is imperative that these models use large quantities of real-world data which is valid and consistent. Therefore, many classified and unclassified databases are maintained at RAND as input to simulation models. As part of our research, we observed and analyzed the use of these "public" databases as input to specific simulation models.

When simulation developers attempt to use these databases, they face some major obstacles. Most of the public databases are acquired from federal agencies which distribute data to a wide variety of clients and customers. When the databases arrive at a client's site, they are generally organized as record-oriented flat files that are subsequently "relationized" and loaded into a DBMS. However, the resulting relational schema is not designed using established database design or modeling principles and is not developed with the assistance of any domain experts. Consequently, semantic integrity constraints that should apply to the public databases are rarely expressed in the relational schema organization or reflected in the data instances. In addition, little documentation is provided with the public databases, and many data values are missing, inconsistent, and erroneous.

At RAND, these public databases are maintained in Ingres. Upon closer examination of database usage, we discovered that the relationship between the Ingres databases and their simulation-specific counterparts is not at all isomorphic. Most of the public databases contain more data than is necessary for a particular study; therefore, it is common for a simulation builder to extract a subset of the Ingres databases for use as input to a specific simulation model. Furthermore, modelers and analysts require data which is tailored to their own specific simulation needs. Their requirements usually entail a combination of transformations to the Ingres databases to derive a database with the desired profile.

During this derivation process, the semantics of the data play a major role in the integration and abstraction operations performed by the simulation builder. Unfortunately, DBMS query languages cannot easily or directly express these transformations. The database manipulations are usually performed through the joint efforts of an application expert and a database specialist. The application expert uses his or her domain expertise to decide how data records should be integrated and abstracted; the database specialist contributes by providing knowledge about database operations to achieve the desired view. Furthermore, many of the required operations cannot be performed within the DBMS query language and also require the services of an applications programmer.

2.1. Database preparation activities

By observing modelers interacting with relational databases, we have identified three distinct cognitive phases that are critical in composing simulation databases: *mental modeling and synthesis*, *conceptual retrieval*, and *semantic validation*. In the following subsections, we discuss the limitations of interactive DBMS facilities and how they hamper each phase of database interaction. Our objective was to remedy these deficiencies by providing an interactive environment supporting database preparation processes.

2.1.1. Mental modeling and synthesis

When users are presented with the task of browsing through a database, they tend to preview the data in a fashion which helps them mentally abstract major concepts and relationships. The first phase of this process is usually scanning the relational tables and attribute names to arrive at a central organizing theme. For someone unfamiliar with the specific relational database, this activity is difficult because attribute names are non-intuitive acronyms listed in a data dictionary with no description of meaning or usage. After a user tries to glean an overall organization of the relational structure, he or she begins looking at rows of values in the relational tables. The relational model does not naturally represent hierarchical concepts; therefore, users frequently search through data and schema hoping to find some hierarchical organization as a basis for abstracting the flat relational tables. Furthermore, most data is encoded and unformatted, providing little evidence that their mental model of the structural organization is valid and consistent. By iteratively looking at the relational structure and selected data values, a user begins to synthesize a conceptual image of the entities and relationships represented in the database and how they map to the necessary simulation concepts.

Although query languages allow flexibility in searching and selecting records based on syntactic pattern matching and efficient indexing techniques, they do not provide an overview or general presentation of the data. If a potential user is familiar with a database and is an experienced DBMS user, then it is much easier to browse through a database in search of specific concepts and entities. However, for the casual user there are few tools or friendly environments to support this modeling and synthesis process. For example, if a simulation builder needs information about the 67th Armor Division, he or she may approach this query by searching all tables for the string "67th Armor Division". It is unlikely that this query will retrieve any useful information. First, "67th Armor Division" is probably abbreviated or encoded so a syntactic search may not produce any matches. Second, there are many different kinds of information that a user may desire about an armor division, such as the general characteristics of the 67th Division, or the subordinate units which are commanded by the 67th Division. A simple text search, however, would not provide any explanation of what is retrieved, only specific data values.

2.1.2. Conceptual retrieval

After a user has gained some familiarity with the organization, structure, and content of a particular database, he or she must determine what data to retrieve for deriving a specific simulation database. The user compares his or her mental model of what is in the database with a conceptual profile of the desired data. Based on this comparison, the user must retrieve those relational entities which map onto the desired conceptual profile. A significant factor which simulation builders consider is the granularity or resolution of the information. Most often, the public databases represent a finer resolution than is needed for the resulting database. Integrating and aggregating data elements play a major role in composing a simulation database.

Users would like to access and retrieve data from public databases whose values collectively represent conceptual entities or relationships. However, query languages provide only a microscopic view of data entities and elements. To derive an entity with the desired profile, a user must translate the profile into standard DBMS selection, projection and join operations. For example, if a user wants to retrieve all "reconnaissance" aircraft, he or she must mentally compose a semantic description for the concept of reconnaissance. Next, the user must map the semantic description onto the attributes and data available in the public databases. Finally, a DBMS query is constructed which integrates data from various sources, retrieving those items which correspond to the concept of reconnaissance. Similarly, to derive a value for "firepower" associated with an airbase, it is necessary to access and aggregate a number of variables upon which firepower is dependent.

Although many of these capabilities can be performed by programs using an embedded data manipulation language, we should not expect casual DBMS users to become DBMS experts simply to browse through the data and retrieve relevant conceptual entities. View mechanisms and embedded data manipulation languages are similarly geared toward interfacing application programs with the database and ignore the needs of interactive users.

2.1.3. Semantic validation

The final activity performed when constructing a simulation database is to validate the correctness of the structure and content of the derived database. In many cases, the data which has been selected may not be consistent or correct. Numeric cross tabulations may be incorrect if only a subset of the database is retrieved. Existence dependencies between entities may also need to be verified. For instance, a user may want to enforce a constraint stating that if long range bombers are located at an airfield, then the airfield must have at least one concrete runway. In addition, the simulation developer may want to add additional constraints on the derived database which did not hold for the public databases.

We have observed this validation process being carried out jointly by a database and domain expert. This task is usually performed by manually searching through data records, looking for suspect or errorful values. Often, simply the presence of a data record will trigger, in the mind of the domain specialist, a condition or constraint which should be considered in the simulation database. Augmenting the resulting database is also common when necessary data is not available from the public databases.

3. IID capabilities

Traditional data dictionaries are used for defining DBMS entities, generating reports, and expressing transactions, but are not suitable as an interface between an interactive user and a DBMS [Alle82]. Our intelligent information dictionary fills this need by addressing the three phases of database preparation discussed above. IID supports *explanation and browsing*, *customized data manipulation*, and *interactive consistency checking* by combining domain knowledge with relational DBMS knowledge. Object-oriented knowledge bases in IID represent both the constructs of a relational database and domain specific knowledge acquired from an application specialist. IID is implemented in Franzlisp Flavors running on a Sun Microsystems workstation. The dictionary communicates with the Ingres relational DBMS, also resident on a Sun machine. The application database we are utilizing to test IID is an air order of battle (AOB) database representing air resources such as airbases, runways, and aircraft. The examples shown in the following sections are derived fron this database and other similar military databases. Details of the IID architecture are presented in Section 4.

3.1. Explanation and browsing

Explanation and browsing is enabled in IID by presenting an extensive collection of metadata to users to guide them through the maze of relations and attributes. Metadata in our information dictionary does not refer strictly to information needed by the DBMS, such as data type and field length. Rather, it refers to semantic information about the data which users rely on when making decisions about how entities relate to each other, and whether the data is relevant to their application. Figure 1 shows the user interface for browsing through relations and columns. Much of the information show in Figure 1 is maintained strictly within the information dictionary without accessing the relational databases. In this example the user is viewing the column names for the "aircraft" relation, and the column "btype" is further described. The allowable values for btype are expressed as "Value constraints". The items in this enumerated list are mouse sensitive and can be further described as we have shown for the value "BC".

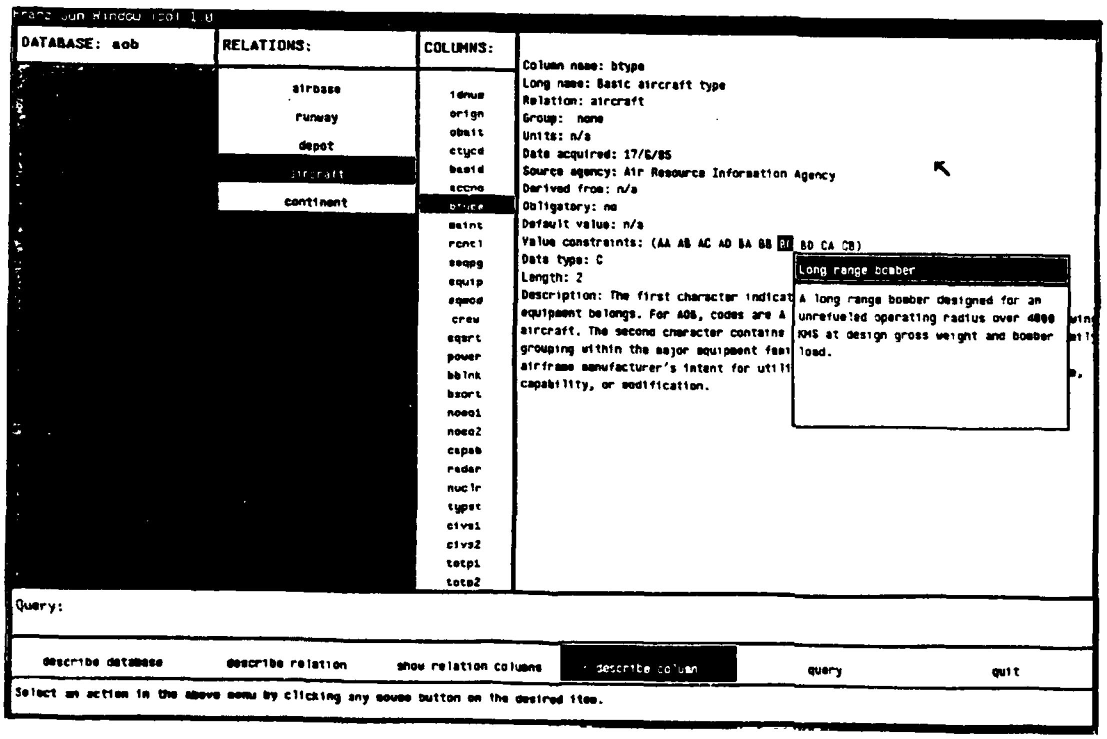

Figure 1: IID explanation and browsing

Values for column descriptors such as "Date acquired", "Source agency", and "Derived from", are represented and maintained in the information dictionary. Therefore, when a new version of the AOB database is loaded, the "Date acquired" field reflects this new information. The descriptors "Obligatory", "Default value", and "Value constraints" not only provide explanatory information but also have associated procedures which interactively validate data instances. These capabilities are further discussed in the section describing consistency checking.

Another feature useful for browsing through an unfamiliar database is IID's *verbose* mode. When verbose mode is enabled, encoded DBMS output is expanded into its full textual name or identifier. Similarly, input to the DBMS through IID can contain fully expanded abbreviations. For example, in a traditional query language, a user must know the country code for France in order to retrieve all aircraft located in France. With IID's verbose mode, a user can submit the query:

retrieve (aircraft.idnum) where country = "france"

Although "country" is not a valid column name and "France" is not an allowable value for the column name "ctycd", IID preprocesses the input and submits to Ingres the query:

retrieve (aircraft.idnum) where aircraft.ctycd = "fr"

Similarly during output, any country codes will be expanded to their full country name.

These browsing capabilities help a user interact with a DBMS in a more natural fashion and distance the user from the unintelligible *codified* aspect of databases maintained by a DBMS. Other research efforts are also addressing the issue that DBMS interfaces are unsuitable for casual users. The Rabbit system [Tou82] aids user interaction through an iterative process of *query reformulation.* Both IID's browsing capabilities and Rabbit's retrieval by reformulation attempt to facilitate the user's understanding of instance data.

3.2. Customized data manipulation

One major obstacle facing interactive users is the lack of encapsulation facilities for grouping individual data values and referencing them as a single semantic concept. Users emulate conceptual retrieval by repeatedly navigating through relations to retrieve the desired data. Instead, they would like to express a combination of attributes and values as a *customized conceptual package* and subsequently refer to that concept as a single entity. Three IID features encourage customized data manipulation: *view templates*, *smart joins*, and *aggregation functions*.

3.2.1. View templates

View templates allow a DBMS user to build a template or profile for a particular concept, and refer to that template for data access. IID translates a view template into an acceptable query and submits it to the DBMS. In Figure 2, we show the template describing a *reconnaissance* aircraft. Although the concept of ***mission*** is not explicitly represented in this database, the user's notion of a reconnaissance mission translates into characteristics represented in the database such as aircraft capabilities and radar equipment. In this example, "mission" can be regarded as a *virtual* column for aircraft and "reconnaissance" can be viewed as a virtual value for the column.

The retrieve command shown in the "Query:" window of Figure 2 indicates how the user would express a *templated* query. With this capability, domain specialists can refer to database entities in their own terminology and can customize the meaning of application-specific concepts and relationships. The window interface also allows a user to view the template description as we have shown in the "TEMPLATE:" window of Figure 2.

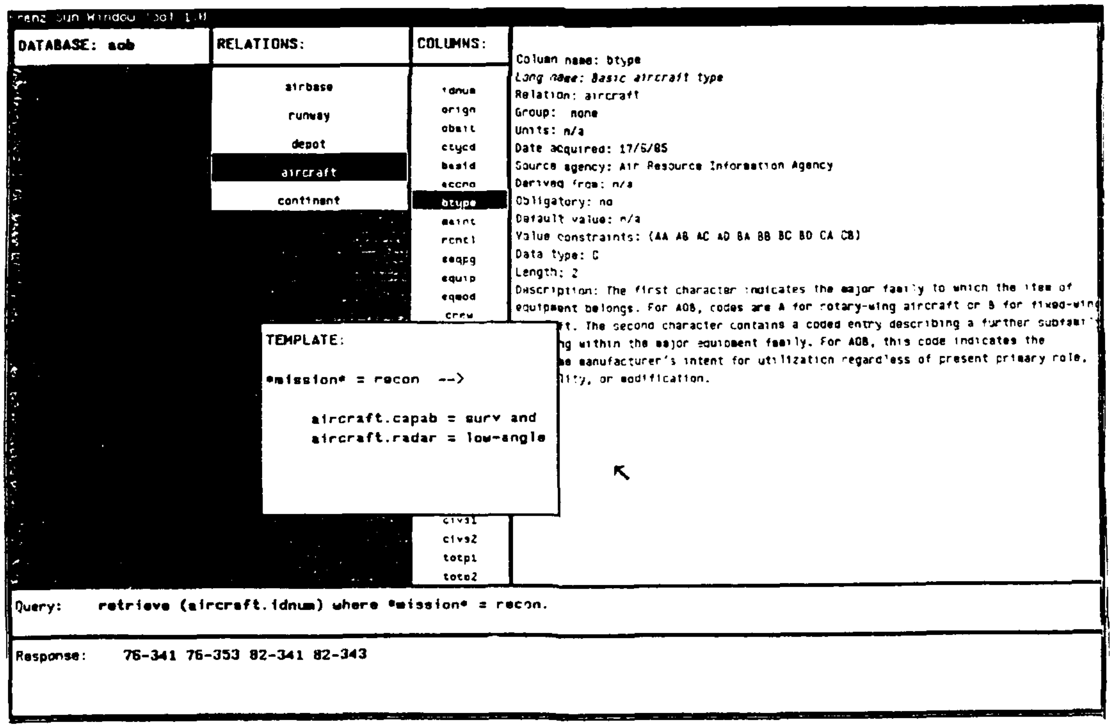

Figure 2 A view template for reconnaissance

3.2.2. Smart join operator

Expressing a join in a relational query language requires a user to navigate through foreign key attributes. In IID, a smart join operator uses the information stored in the dictionary to recognize implicit links between relations. For instance, in Figure 3 a user wishes to retrieve all aircraft of a certain type that are located in Europe. However, aircraft are not recorded by continent; rather, they are organized by country. In Quel, this query would be expressed as:

retrieve (aircraft.idnum) where aircraft.btype = "AC" and
aircraft.baseid = airbase.baseid and
airbase.ctycd = continent.ctycd and
continent.name = "europe"

Using IID's join feature, the user need only submit the retrieve command shown in the "Query:" window of Figure 3. During IID query preprocessing, the dictionary refers to its domain specific knowledge about aircraft and airbases, and to its knowledge of relational joins. Join fields may be specified with the database dictionary information or may be expressed interactively by a user. IID translates the user's query into the equivalent Quel command and submits it to Ingres. The window interface also depicts the relationship between "aircraft", "airbase", and "continent", providing a map of the correspondence between relations. This display feature is also convenient for browsing through the network of relations.

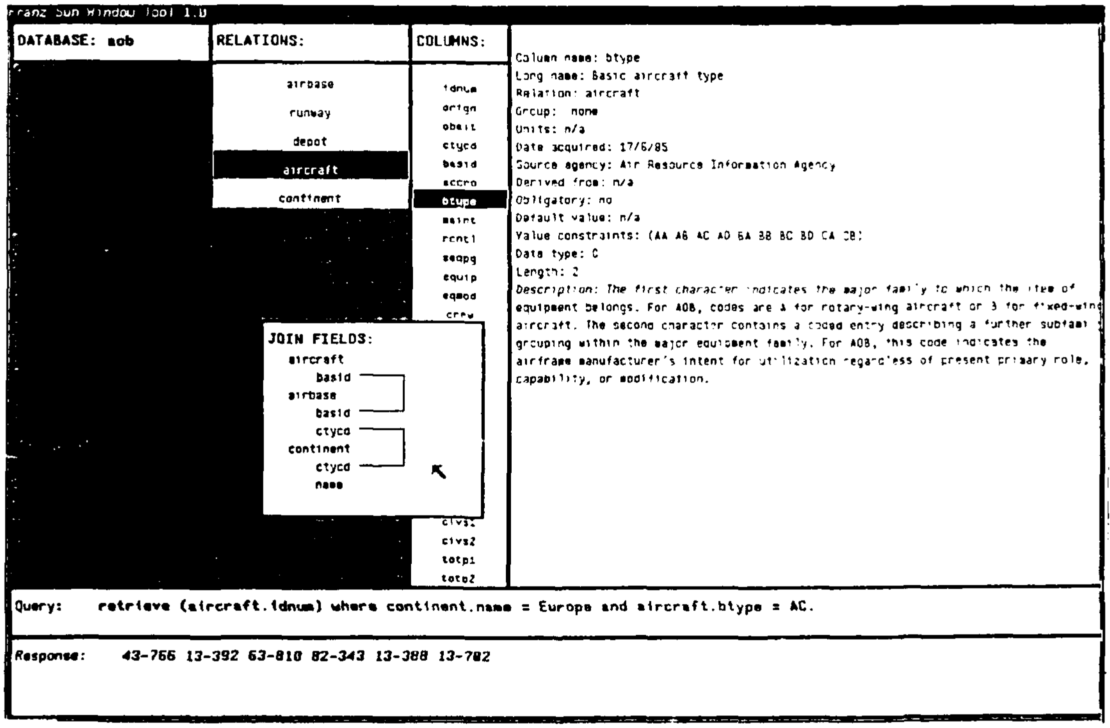

Figure 3 A smart join operation

3.2.3. Aggregation functions

Aggregation functions are useful for abstracting a detailed database and retrieving data at the desired level of granularity. Built-in DBMS aggregate functions are limited to operations such as count, sum, and average. IID's aggregation functions allow a user to express a domain-specific aggregation and use that specification for interactive queries. Consider the query shown in Figure 4. In this example, the user is interested in computing the *firepower* associated with airbases. In the AOB database, however, firepower is not recorded for airbases but rather it is stored with *missile depots*. An IID aggregation function allows a user to specify how firepower should be computed, and to subsequently invoke the pre-specified declaration. In this way, the user has the flexibility to interactively derive appropriate data. Aggregation functions, like view templates, enable users to construct virtual columns which are automatically expanded by IID. The interactive query in Figure 4 is translated into a Quel query of the form:

retrieve (airbase.baseid, firepower=avg (depot.firepower
by depot.baseid))
where airbase.baseid = depot.baseid

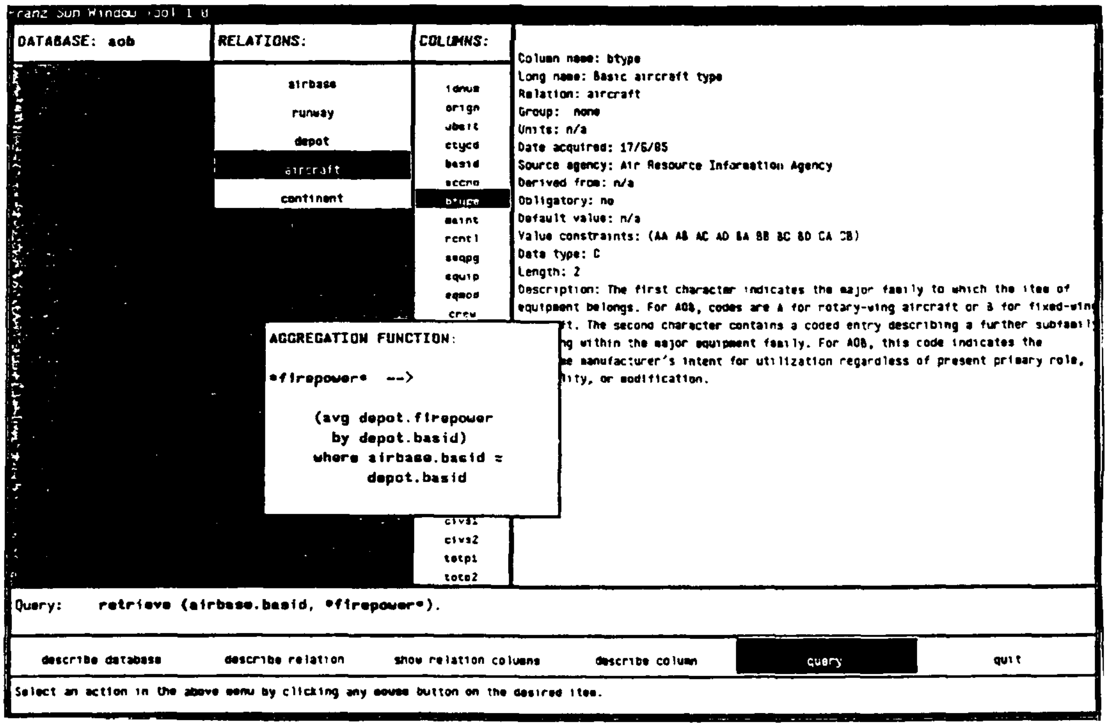

Figure 4 An aggregation function for firepower

Aggregation functions, view templates, and a smart join operator supply the user with a more comprehensible view of the database without forcing the user to review the details of each record. We envision users constructing libraries of view templates and aggregation functions. These stored declarations can be accessed and viewed by other users thereby creating multiple perspectives. *Soft selection criteria* [Mart86] is another area which relaxes the rigidity of DBMS query languages by allowing subjective or varying selection qualifications. Similar research has been conducted for statistical databases where aggregation is a predominate function in database queries [Chan81].

3.3. Interactive consistency checking

Many knowledge base management systems and intelligent database systems are providing facilities to automatically verify semantic constraints through the use of triggers and allerters [Ston85]. For interactive database users who are deriving *personal* databases from large public databases, conventional methodologies for constraint specification and enforcement do not apply. Conventional constraints must be built into the data dictionary by a database administrator and data modeler. Instead, users would like to express new value and structural constraints, and modify those declared for the public databases. The semantic validation capabilities available in IID allow interactive "scrubbing" of data values in the public databases which may be incorrect or inconsistent with the resulting derived database. The information dictionary maintains knowledge about obligatory fields, default values, and value and structural constraints. For instance, if the number of aircraft owned by a squadron is set to "20", the user would be notified that "20" is an incorrect value because of a domain rule stating that the number of aircraft owned by squadron must be a multiple of 6. A user may also want to express an existence constraint of the form:

If there is a nuclear launching site on an airbase,
then there must be at least one nuclear weapons depot on that airbase.

Validation mode in IID supports both value constraints and functional dependencies. If the above rule has been entered and validation mode is enabled, then a validation procedure is invoked whenever the user retrieves a nuclear launching site. If a nuclear weapons depot does not exist in the derived database, the user would be notified that a constraint rule has been violated.

Constraint management is receiving much attention in the context of expert database systems and knowledge base management systems. In many cases, researchers are advocating constraint specification, propagation, and satisfaction as an underlying formalism driving the entire processing of the system [Shep86]. In IID, however, consistency checking is approached from a very localized perspective, that is, a user's derivation of application specific databases. IID's validation procedures report invalid data but currently make no attempt to correct inconsistencies.

4. IID architecture

The IID software system is comprised of three major processing components: the Lingres interface, IID object-oriented dictionary framework, and user interface. These components are shown as shaded modules in Figure 5. Domain dependent information reside in at least three data and knowledge bases: the relational database maintained in Ingres, the "public" knowledge base, and one or more "private" knowledge bases. Figure 5 depicts these "domain specific" entities as dotted modules. Explanation and browsing, verbose mode, and validation mode are currently operational in IID. Limited functionality for aggregation functions, smart joins, and user templates has been tested and general versions are currently under development. Design and implementation of an extended user interface is also underway.

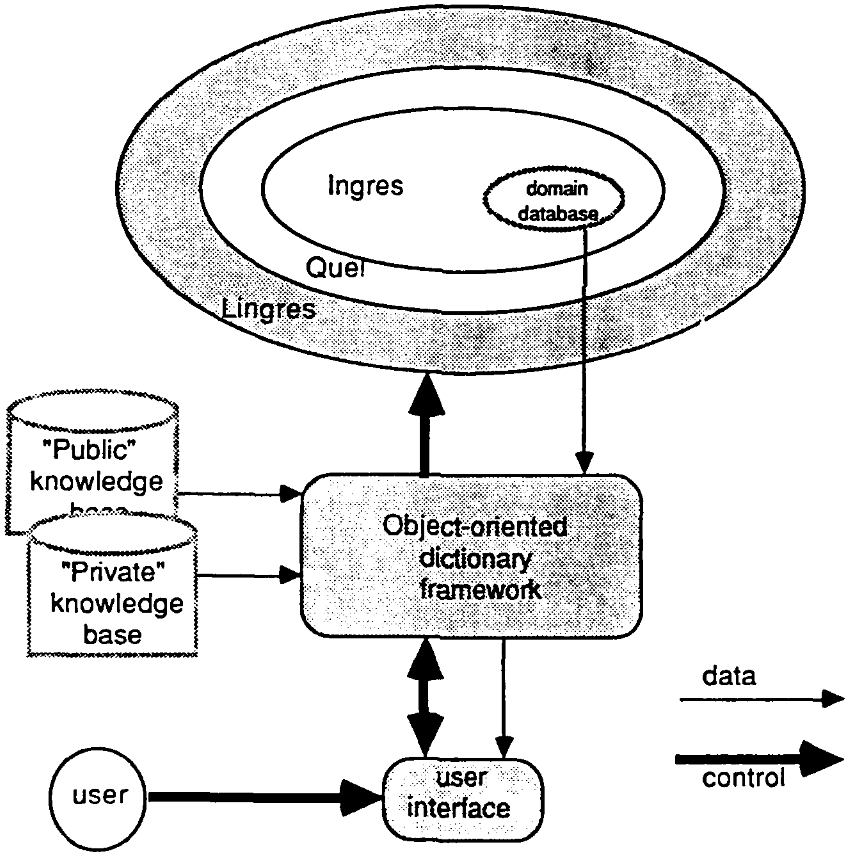

Figure 5 IID system architecture

We implemented IID's user interface on top of a Sun View windows package in Franzlisp. Our initial version of the user interface was designed to give the user the full range of functionality provided by IID at the expense of overloading the user with too many options. We are experimenting with various designs and organization of windows, menus, and input procedures to maximize the capabilities of IID while minimizing the cognitive overhead incurred by hierarchies of menus and windows.

Commands issued to IID through the user interface are passed on to IID's object-oriented dictionary framework. We implemented this module as a domain-independent and DBMS-independent environment for reasoning about relational database entities. The dictionary framework incorporates general knowledge about DBMS concepts such as relations, attributes, and joins. In addition, the framework accesses domain-specific information necessary for IID processing. During IID user interaction, the dictionary framework combined with domain information, such as AOB data and knowledge bases, enables facilities like verbose mode, consistency checking, and automatic join operations by reasoning about database structures and domain entities.

Much of IID's processing translates a user's query into a syntactically and semantically valid Quel query. This translation is similar to the programming language concept of *macro expansion* where a succinct declarative expression is replaced by a detailed procedural expression. IID constructs a Quel command (semantically equivalent to the user's input query) and submits it to Ingres through the Lingres interface. Data instances are returned to the user following IID query postprocessing.

Three separate sources of information comprise the domain dependent components of IID. One data repository is the public relational database maintained in Ingres. The AOB database we are utilizing to develop and test IID contains three main relations of approximately 2000, 7700, and 11000 tuples, and the number of fields per relation ranges from 25 to 36. This database is the source of value data retrieved in response to IID queries.

A "public" knowledge base corresponding to a domain database is one of two knowledge sources representing the semantics of the relational database. The information retained in this knowledge base can be regarded as the *default* semantics applicable to the domain database. Development of this component for the AOB database resulted from a data modeling effort which proceeded in parallel with the design and implementation of IID's processing components. The knowledge base is organized as objects relevant to AOB entities such as aircraft and airbases. Extensive descriptional information to support IID's explanation facilities was acquired from AOB domain specialists and extracted from various sources of documentation. The second knowledge source, a private knowledge base, represents semantic information derived by a user. This knowledge base augments the public knowledge base and stores aggregation functions and view templates for a particular simulation model or user. We envision many private knowledge bases representing different views of the domain database. During our initial developement effort, however, we have been experimenting with a single private knowledge base.

The Lingres facility we developed allows access to an Ingres database from Flavors and Lisp, and offers the full functionality of Quel's retrieve, create, delete, destroy, and append commands. Lingres maintains only the schema and dictionary data that Ingres itself supports. These capabilities are similar in functionality to those offered by Kee Connection, an expert system tool interface [Inte87]. For the purposes of modularity and future development, our goal for Lingres was merely to mimic the functionality of Ingres, i.e.,"Ingres in Lisp". In the future, changing to Common Lisp (or another Lisp) will affect only the "Lisp to C" connection; supporting SQL will only require new parsing routines in

Lingres; and changing to another relational DBMS, e.g., Oracle, would simply mean replacing Ingres system calls with calls to Oracle system routines. One of the most time consuming operations of IID is communicating with Ingres; therefore, Lingres also improves IID efficiency by duplicating Ingres metadata, thereby reducing communications with Ingres.

5. Related research efforts

Similar work addressing browsing and explanation is often refered to as *metadata management*. Mark and Roussopoulos [Mark86, Mark87] approach metadata management through *self-describing* data models. They are developing capabilities, similar to IID, to initially browse through the schema to learn about the database and then proceed to access the data. They are applying their model to facilitate standardized information interchange. In this application, however, they are not dealing with semantic aspects of the data, and therefore have not incorporated domain specific knowledge for explanation and validation.

Information Resource Dictionary Systems (IRDS) have also been the subject of a considerable amount research [Dolk87, Kers83, Nava86]. In the past, IRDS were considered primarily as a design tool for information modeling and database design. Only *active* data dictionaries were utilized during batch DBMS operation or real-time transaction processing. In [Gold85] Goldfine describes the National Bureau of Standards specifications of an IRDS system. This standard specifies a kernel set of basic data dictionary capabilities plus a collection of independent optional modules. So far, three additional modules have been specified dealing with security, application program interface, and documentation. The emphasis on program interfaces, and neglect of interactive tools is evident in the specification. However, we are seeing efforts extending the kernel IRDS specification to support interactive environments [Koss87].

The scope of an IRDS system embodies the major activities, processes, information flows, organizational constraints, and concepts of an "Enterprise Model". Because interactive use of a DBMS and data dictionary have not been feasible until recently, traditional information management processes do not include casual users exploring a database, deriving new databases, or sharing *personal* databases. We expect that the functionality offered by IID will become necessary as the use of interactive information systems proliferates.

6. Conclusions and future work

In this paper we have discussed a development effort and resulting software for improved interactions between a database user and a relational DBMS. IID is targeted to aid a casual database user who is familiar with the database domain but is not an experienced DBMS user. One of our research goals was to extract domain-specific information from a simulation expert and incorporate it as knowledge in the information dictionary. Another goal was to represent knowledge about relational database operations, such as join operations, to support query construction. Explanation and browsing, customized data manipulation, and consistency checking are the main processes supported by the IID interactive environment. Our initial studies indicate that IID facilities augmented with domain and database knowledge will significantly streamline the interactive preparation of simulation databases.

Development of IID grew out of a larger project which is addressing the use of large heterogeneous databases in object-oriented simulation systems. We have recognized that the preparation phases of mental modeling and conceptual data manipulation stem from attempts to view a flat relational database as an object-oriented hierarchy of simulation entities. IID strives to present the mapping between relations and domain entities more explicitly.

In the short term, we will be expanding IID's facilities to include spatial presentation and aggregation. In many public databases, location of entities is a major determinate in whether or not the entity is included in a simulation database. We plan to extend the explanation facilities so that spatial data can be quickly plotted on a geographical map, and data points on the map can be easily accessed and aggregated.

Configuration management of both public and simulation databases is another desirable feature. Users would like to be notified if any of their simulation data has been invalidated by a new version of the public databases. Furthermore, they hope to be able to pose queries about the changes that were enforced by a new version, such as: What is the difference between the old and new versions of the F-14 aircraft data? To support this feature, it is necessary to track and log the derivation of any simulation database and reason about operations which produced the resulting data.

In parallel with IID development, we will be augmenting our simulation knowledge base with metadata and constraint information related not only to the relational aspects of the data, but also to the object-oriented schema of the data. Our long term objective is the use of IID as an active information dictionary within an object-oriented simulation language. In this role it will provide a dynamic communication channel between a object-oriented semantic schema and the corresponding relational instances of many diverse public databases.

References

[Alle82] Allen, F. W., Loomis, M. E., and Manning, M. V., "The integrated dictionary/directory system," *ACM Computing Surveys* **14**(2), pp.245-286 (June 1982).

[Chan81] Chan, P. and Shoshani, A., "Subject: A directory driven system for organizing and accessing large statistical databases," pp. 553-563 in *Proceedings of the seventh international conference on very large Data Bases*, Cannes, France (1981).

[Curt81] Curtice, R.M., "Data dictionaries: An assessment of current practice and problems," pp. 564-570 in *Proceedings of 7th conference on very large data bases*, Cannes, France (September 1981).

[Dolk87] Dolk, D. and II, R. Kirsch, "A relational information resource dictionary system," *Communications of the ACM* **30**(1), pp.48-61 (January 1987).

[Gold85] Goldfine, A., "The information resource dictionary system," pp. 114-122 in *Proceedings of the fourth international conference Entity-Relationship approach*, Chicago, IL (October 1985).

[Inte87] *Intellinews* 3(2), Intellicorp (January 1987).

[Kers83] Kerschberg, L., Marchand, D., and Sen, A., "Information system integration: A metadata management approach," pp. 223-239 in *Proceedings of the fourth international conference on Information Systems*, Houston, TX (1983).

[Koss87] Kossman, R., "An active information resource dictionary," in *Proceedings of Ingres user association meetings*, San Francisco, CA (April 1987).

[Mark86] Mark, L. and Roussopoulos, N., "Metadata management," *Computer* **19**(12), pp.26-35 (December 1986).

[Mark87] Mark, L. and Roussopoulos, N., "Information interchange between self-describing databases," *Data Engineering* **10**(3), pp.46-52 (September 1987).

[Mart86] Martin, D., *Advanced database techniques*, The MIT Press, Cambridge, MA (1986).

[McCa82] McCarthy, J.L., "Metadata management for large statistical databases," in *Proceedings of 8th conference on very large databases* (September 1982).

[Nava86] Navathe, S. and Kerschberg, L., "Role of dictionaries in information resource management," *Information and Management* **10**(1), pp.21-46 (January 1986).

[Shep86] Shephard, A. and Kerschberg, L., "Constraint management in expert database systems," pp. 309-331 in *Expert database systems*, ed. L. Kerschberg, Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA (1986).

[Ston85] Stonebraker, M. and Rowe, L.A., "The design of POSTGRES," Memorandum No. UCB/ERL 85/95, University of California, Berkeley, Berkeley, CA (November 15, 1985).

[Tou82] Tou, F. N., Williams, M. D., Fikes, R., Henderson, A., and Malone, T., "Rabbit: An intelligent database assistant," pp. 314-318 in *Proceedings of the third annual national conference on artificial intelligence*, Pittsburg (1982).